

EV316936543

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Method and Apparatus for Configuring A Server

Inventors:

Kirk Soluk

Everett McKay

Hitesh Raigandhi

Yang Gao

Praerit Garg

ATTORNEY DOCKET NO. MS1-1575US

1 **TECHNICAL FIELD**

2 The systems and methods described herein relate to assisting a user in
3 configuring one or more servers based, in part, on a user's response to various
4 requests.

5
6 **BACKGROUND**

7 Different types of servers are used to perform a variety of tasks in, for
8 example, a network environment. Example servers include file servers, print
9 servers and web servers. A file server is a computing device and a storage device
10 that stores files. Users on a network can store files to the file server and retrieve
11 files from the file server. A print server is a computing device that manages one or
12 more printers coupled to the print server or coupled to a common network. A web
13 server is a computing device coupled to a network (such as the Internet) that
14 delivers (i.e., serves up) web pages. A web server has an IP address and, in some
15 cases, a domain name. Servers are often dedicated such that they do not perform
16 other tasks outside their server tasks. Certain servers may be capable of
17 performing multiple tasks or functioning as multiple servers simultaneously (such
18 as a file server and a print server).

19 Initially, a server needs to be configured to perform the various functions
20 desired by a network administrator or other operator. Configuration of the server
21 may also define certain operating characteristics, such as what types of users can
22 access the server, how the server handles various types of requests, and what types
23 of data should be encrypted by the server.

24 In existing systems, network administrators or other operators are
25 responsible for configuring servers. Often, this configuration requires a significant

1 amount of understanding and low-level control of the various operating
2 characteristics. These manual tasks are time-consuming and may be redundant if
3 the network administrator is configuring multiple servers that perform similar
4 functions. In these situations, the “quality” of the server configuration is
5 dependent on the knowledge and skill of the network administrator. If the network
6 administrator is inexperienced at configuring servers, the server may not be
7 configured in the best possible manner. Improper or inefficient server
8 configuration may cause problems, such as poor server performance or inadequate
9 server security.

10 Accordingly, there is a need for improved techniques for configuring one or
11 more servers.

12 13 **SUMMARY**

14 The systems and methods described herein configure a target server based
15 on knowledge of roles to be performed by the target server and based on a user’s
16 response to various questions regarding the target server. In one embodiment, a
17 process identifies a role associated with a target server. The process then
18 identifies one or more services associated with the role and identifies one or more
19 ports associated with the role. The identified services and ports associated with
20 the role are presented to a user, who is requested to select among the identified
21 ports associated with the role.

BRIEF DESCRIPTION OF THE DRAWINGS

Similar reference numbers are used throughout the figures to reference like components and/or features.

Fig. 1 is a block diagram illustrating an example architecture capable of configuring one or more servers.

Fig. 2 is a flow diagram illustrating an embodiment of a procedure for configuring a server using the architecture of Fig. 1.

Figs. 3A and 3B illustrate data contained in a portion of an example knowledge base.

Fig. 4 is a flow diagram illustrating an embodiment of a procedure for configuring a server.

Fig. 5 illustrates an example of a Runtime XML file associated with one or more servers.

Fig. 6 is a flow diagram illustrating an embodiment of a procedure for masking role details from a user unless a user requests to see the role details.

Fig. 7 is a flow diagram illustrating an embodiment of a procedure for allowing a user to select services and ports associated with a particular role.

Fig. 8 is a flow diagram illustrating an embodiment of a procedure for determining how to handle services and/or ports associated with a target server that are not defined in a knowledge base.

Figs. 9A and 9B illustrate an example of an Output XML file associated with one or more servers.

Fig. 10 illustrates an example of a computing environment.

1 **DETAILED DESCRIPTION**

2 The systems and methods described herein allow an administrator (or other
3 user) to configure one or more servers based on functional and deployment
4 requirements solicited by a user interface. When configuring a target server, the
5 systems and methods described herein access data contained in one or more
6 knowledge bases along with user-supplied input regarding how the target server
7 will be used. The knowledge bases contain information regarding operating
8 system and application parameters that need to be configured or analyzed from a
9 security perspective. Based on information obtained from the knowledge bases,
10 information regarding the target server and one or more roles identified by a user,
11 the systems and methods present the user with various choices, such as which
12 ports to activate on the target server. These choices are limited to those that are
13 relevant based on the information obtained from the knowledge bases, the
14 information regarding the target server, and the roles identified by the user.

15 Various examples discussed herein refer to one or more servers. As used
16 herein, a “server” is any computing device capable of performing functions
17 defined by a particular server type. As discussed herein, a computing device may
18 need to be configured to perform one or more roles, such as a file server role.
19 Certain tasks associated with the file server role may include retrieving files,
20 storing files and providing files to requesting devices. In particular embodiments,
21 a “server” may perform multiple roles.

22 Particular examples discussed herein refer to configuration of a single
23 server. However, the systems and methods discussed herein can be applied to the
24 configuration of any number of servers. For example, if two servers are to be
25

1 configured in similar ways, the two servers can be configured by creating a
2 transform that can be deployed to configure both servers.

3 Specific examples discussed herein refer to Extensible Markup Language
4 (XML) files. XML represents one possible language that may be used with the
5 systems and methods discussed herein. In alternate embodiments, any other
6 language or file storage format can be utilized.

7 As discussed herein, a particular server may be configured to perform one
8 or more roles. A role defines, for example, a set of functions or tasks that the
9 server performs. Example roles include a file server, a print server, a web server,
10 a domain controller, a DNS server, an SMTP server and a terminal server. A file
11 server role, for example, receives files, stores files and responds to file access
12 requests. Various security parameters are associated with the different roles. For
13 example, roles may have associated security parameters relating to account login
14 policies, event log settings, port filtering policies, Internet Protocol Security
15 (IPSec) policies and security-related registry values. Additional details regarding
16 roles are provided below.

17 Fig. 1 is a block diagram illustrating an example architecture 100 capable
18 of configuring one or more servers. A pre-processor 102 is coupled to a target
19 server 104. Target server 104 is a server being configured or re-configured. Pre-
20 processor 102 obtains information from target server 104, such as how the target
21 server is currently configured, the types of roles that the target server is able to
22 perform, etc. Pre-processor 102 also retrieves information from a knowledge base
23 106. Knowledge base 106 may also be referred to as an “unprocessed knowledge
24 base”. Knowledge base 106 is an Extensible Markup Language (XML) file that
25 identifies operating system and application parameters that need to be configured

1 or analyzed from a security perspective. The information in knowledge base 106
2 is presented in functional terms that are more easily understood by network
3 administrators who are not necessarily server or security experts. Knowledge base
4 106 also contains directives utilized by the user interface that help determine the
5 default values rendered by the user interface under various circumstances such as
6 the current configuration of the target server 104 and previous choices made by the
7 user. Additionally, knowledge base 106 defines mappings between abstractions
8 (e.g., roles) defined in the user interface and the underlying settings (e.g.,
9 services).

10 In a particular implementation, target server 104 is not actually configured.
11 Instead, target server 104 is merely used to assist a user in creating a policy for a
12 system similar to the target server. The target server assists the user by causing the
13 filtering out of information in the knowledge base that doesn't apply to the
14 environment for which the user is creating a policy. In an alternative embodiment,
15 a target server is not used to generate a policy. In this embodiment, the user has
16 access to all data in the knowledge base.

17 Pre-processor 102 creates a run-time specific knowledge base, which is
18 referred to in Fig. 1 as a Runtime XML 108. Runtime XML 108 may also be
19 referred to as a "processed knowledge base". Runtime XML 108 is created by
20 comparing the raw knowledge base data 106 to the current state of target system
21 104. By pre-processing the knowledge base data 106, a user interface application
22 110 (which receives Runtime XML 108) is able to render more appropriate
23 information and start with more accurate details. For example, if particular roles
24 cannot be supported by target server 104, Runtime XML 108 can indicate those
25 unavailable roles such that user interface application 110 does not offer those

1 unavailable roles to the network administrator for selection by default.

2 Additionally, pre-processor 102 identifies other information that is not contained
3 in knowledge base 106, such as network configuration information (e.g., network
4 interface controllers and IP addresses on the target server). This information is
5 provided to user interface application 110 and is helpful in providing the correct
6 default choices to a user. For example, if a role is already selected on the target
7 server, the user interface application 110 will present that role as being selected by
8 default.

9 In a particular embodiment, Runtime XML 108 contains all of the data
10 from knowledge base 106, but the data elements are “tagged” to indicate whether
11 the particular elements (e.g., particular roles) are supported by target server 104.
12 An example tag is: <Satisfiable> TRUE </Satisfiable> In another embodiment,
13 the data elements that are not supported by target server 104 are filtered out by
14 user interface application 110 such that they are not included in Runtime XML
15 108. This “filtering out” process is accomplished by the tags discussed above.
16 The two tags are <Satisfiable> and <Selected>, which tell the user interface
17 application 110 whether or not the role can be performed by the server. If so, the
18 tags indicate whether or not to select that role by default.

19 Runtime XML 108 can be stored on a storage device, such as a hard disk
20 drive, for later retrieval by user interface application 110 or other applications.
21 This architecture decouples pre-processor 102 from user interface application 110.
22 This architecture allows for free-form XML based data structures as a way of
23 interacting between the configuration engine and the user interface application.

24 Runtime XML 108 is provided to user interface application 110, which
25 obtains additional input about the desired configuration of target server 104 from a

1 network administrator or other individual. Using the data in Runtime XML 108,
2 user interface application 110 is able to restrict the options and configuration
3 information requested from the network administrator to those options and
4 configuration information that are relevant to target server 104. For example, if
5 target server 104 is not capable of functioning as a DNS server, then user interface
6 application 110 does not present that option to the network administrator.

7 Although the option is not presented to the network administrator, the role is still
8 available by taking a different “view” of the Runtime XML. Thus, policies can be
9 created without necessarily being connected to the appropriate server.

10 Additionally, if user interface application 110 identifies the operating system (OS)
11 level of target server 104 (e.g., by querying the target server), the knowledge base
12 data can be consulted to determine the set of security levels supported by the
13 system. Thus, user interface application 110 utilizes information from knowledge
14 base 106 and knowledge of the current state of target server 104 to solicit
15 functional server requirements from the network administrator and generate a
16 resulting security policy (shown in Fig. 1 as an Output XML 112). In one
17 embodiment, the user interface application 110 queries the OS version to
18 determine which knowledge base should be used. In this embodiment, there is a
19 primary knowledge base associated with each OS version.

20 In a particular embodiment, two different security levels are available:
21 Maximum and Typical. These two security levels control how the pre-processor
22 102 chooses the default settings that are rendered in user interface application 110.
23 For example, if the user chooses “Maximum” security, the pre-processor will
24 generate a Runtime XML that has minimal roles selected.
25

1 User interface application 110 also guides the user through the creation of a
2 security policy that will provide appropriate security for a given deployment
3 scenario. Guidance is provided by navigating the user through a series of
4 questions designed to solicit the functional requirements that the server needs to
5 provide. Disabling functionality that is not required enhances security by 1)
6 reducing the amount of potentially problematic code that is running, 2) disabling
7 the usage of insecure legacy protocols, 3) limiting the services exposed to
8 untrusted users, and 4) making appropriate operating system and application
9 configuration settings.

10 Output XML 112 is provided to a configuration engine 114 that is coupled
11 to target server 104. Output XML 112 may also be referred to as a “security
12 policy”. Configuration engine 114 (also referred to as a “back-end engine”)
13 provides a framework and a set of components, a transformation module 118 and a
14 configuration module 120, that can transform Output XML 112 generated by user
15 interface application 110 into native scripts and configuration files. These native
16 scripts and configuration files are stored temporarily or permanently on a data
17 storage device 116 coupled to configuration engine 114. The native scripts and
18 configuration files are subsequently applied to target server 104 during a
19 configuration phase. Configuration engine 114 also enables or disables various
20 services, as needed, to implement the roles selected by the user through user
21 interface application 110. Configuration engine 114 also supports rollback and
22 analysis operations in addition to configuration operations. Rollback is a flag that
23 can be passed to a “back-end” transformation or application engine. Rollback is
24 the operation of restoring the system to a previous state. The rollback operation is
25 supported by a “back-end engine” (not shown). A rollback transformation

1 compares the configuration policy to the current state of the system and stores the
2 current values for the changes that would be made if the policy were applied. If
3 desired, these stored current values can subsequently be submitted to the
4 configuration engine to undo a previously applied policy.

5 Analysis operations determine whether a system is in compliance with a
6 particular policy. For example, if an output policy indicates “disable service x”,
7 the analysis operations will generate a warning if service x is enabled. The
8 primary difference between the analysis operations and rollback is their outputs.
9 The rollback output needs to be resubmitted to the configuration engine and the
10 analysis operation output needs to be viewable in a user interface.

11 Transformation module 118 (in configuration engine 114) transforms the
12 data in Output XML 112 into native scripts and configuration files. Output XML
13 112 is typically a high-level policy written in XML. Transformation module 118
14 transforms the high-level policy to native scripts and configuration files that can
15 be understood by target server 104. Configuration module 120 applies one or
16 more policies (defined by native scripts and configuration files) to target server
17 104, thereby configuring the target server. The transformation performed by
18 transformation module 118 is similar to the compilation function performed by a
19 compiler to convert source code into object code for a particular processor
20 architecture. Output XML 112 is similar to the source code and the resulting
21 native scripts and configuration files are created for a particular architecture (the
22 target server).

23 In one embodiment, Output XML 112 represents a high-level policy that
24 will be applied to the target server 104. The high-level policy is transformed into
25 a more specific file customized for target server 104. This high-level policy can

1 be applied to multiple different target servers, in which case a separate customized
2 file is created for each target server. Although the high-level policy is the same,
3 the separate customized files may differ due to differences in the target servers.

4 Although Fig. 1 illustrates a single knowledge base 106, a particular
5 embodiment may include multiple knowledge bases coupled to pre-processor 102.
6 In this embodiment, one knowledge base is the standard knowledge base
7 containing information regarding standard server roles for a particular version of
8 an operating system. Another knowledge base may define roles developed and
9 supported by a third party. For example, a third party may define one or more
10 roles associated with a database server. This use of multiple knowledge bases
11 allows one knowledge base to be updated without affecting the other knowledge
12 bases. Additionally, a user (such as a network administrator) may create their own
13 knowledge base containing one or more role definitions. In one embodiment,
14 these user-created role definitions replace any definitions of roles of the same
15 name in knowledge base 106. In one embodiment, multiple knowledge bases are
16 reconciled together by pre-processor 102 and merged into a single Runtime XML
17 file that is utilized by user interface application 110. Additional information
18 regarding knowledge bases is provided below.

19 Although particular examples discussed herein refer to a target server,
20 alternate embodiments can configure a server without having any previous
21 knowledge of the server's current configuration or capabilities. In these alternate
22 embodiments, the systems and methods may presume that the server is capable of
23 performing any role.

1 The systems and methods discussed herein provide for a “policy authoring”
2 process. This process includes a series of XML transforms (e.g., knowledge base
3 to Runtime XML, Runtime XML to Output XML, and so on).

4 Fig. 2 is a flow diagram illustrating an embodiment of a procedure 200 for
5 configuring a server using the architecture of Fig. 1. Initially, procedure 200
6 retrieves information regarding a target server (block 202). This information
7 includes, for example, the target server’s features, capabilities, and current
8 configuration. The procedure then retrieves information regarding operating
9 characteristics associated with one or more server roles (block 204). These
10 operating characteristics include, for example, services used by various roles,
11 communication ports used by various roles and Internet Information Server (IIS)
12 requirements. The operating characteristics may also include rules for determining
13 whether or not the role is likely being performed by the target server.

14 Procedure 200 continues by generating a Runtime XML document that
15 contains information related to the target server and various roles that servers may
16 perform (block 206). A user interface application then receives the Runtime XML
17 document and obtains additional information from an administrator of the target
18 server or another person (block 208). The procedure then generates an Output
19 XML document (block 210) that contains the high level security policy for the
20 target server or other similarly configured servers.

21 Figs. 3A and 3B illustrate data 300 contained in a portion of an example
22 knowledge base. As shown, various roles, services, ports and protocols are
23 identified and defined. For example, a role named “WEB” has an associated type
24 of “Server”. Three different services are identified: “IISAdmin”, “HTTPFilter”
25 and “W3SVC”. Additionally, two ports are identified: “HTTP” and “HTTPS”.

1 The data 300 also identifies a protocol (TCP). The port "HTTPS" has an
2 associated port number of "443" and uses the TCP protocol. The knowledge base
3 data shown in Figs. 3A and 3B is abbreviated for purposes of explanation. A
4 particular knowledge base may identify and define any number of roles, services,
5 ports, protocols, tasks and other information. A knowledge base entry may also
6 indicate the direction of a required network communication (such as inbound or
7 outbound). For example, a Web Server role requires port 80 to be opened for
8 inbound communication, but a Web Client role would require port 80 to be opened
9 for outbound communication.

10 Certain services are not conveniently abstracted by roles. Services such as
11 snapshot, UPS, performance monitoring, etc. are independent of the "role" the
12 machine is performing. These services can be abstracted by tasks as shown in the
13 following example:

```
14     <Task>
15         <Name> AppMgmt </Name>
16         <DependsOn>
17             <Roles>
18                 <Role>
19                     <Name> DomainMember </Name>
20                 </Role>
21             </Roles>
22         </DependsOn>
23         <Satisfiable> TRUE </Satisfiable>
24         <Selected> FALSE </Selected>
25         <Services>
26             <Service>
27                 <Name> AppMgmt </Name>
28             </Service>
29         </Services>
30         <Ports> TBD
31     </Ports>
32 </Task>
```

1
2 The definition of a task has similar syntax to the definition of a role. Additionally,
3 the semantics for the Satisfiable, Selected, Services and Ports elements are the
4 same as they are for roles.

5 Fig. 4 is a flow diagram illustrating an embodiment of a procedure 400 for
6 configuring a server. Initially, procedure 400 identifies a policy associated with a
7 target server (block 402). The identified policy includes one or more roles that the
8 target server is to perform. Additionally, the identified policy may include one or
9 more roles that the target server is not to perform. Other policies may include one
10 or more roles that the target server is to perform in addition to any roles currently
11 enabled on the target server.

12 The procedure continues by identifying one or more server roles currently
13 enabled on the target server (block 404). A particular target server may have
14 previously been acting as a server and, therefore, has certain server roles enabled.
15 Procedure 400 enables the new roles contained in the policy on the target server
16 (block 406). The procedure then determines whether any of the currently enabled
17 roles on the target server should be disabled (block 408). This determination is
18 based on information contained in the policy associated with the target server,
19 such as “disable all other server roles” or “disable any file server or web server
20 roles”. If one or more roles on the target server need to be disabled, the services
21 and ports associated with those roles are disabled at block 410. Finally, the
22 procedure finishes configuration of the target server (block 412).

23 Fig. 5 illustrates an example of a Runtime XML file 500 generated by pre-
24 processor 102. For the Runtime XML file 500, the pre-processor sets the
25 <satisfiable> element and the <selected> element for each role. “Satisfiability” is

1 determined by whether or not the target system (e.g., the target server) has all the
2 required services for the corresponding role. "Selected" determines the default
3 selections with which each user begins. The pre-processor can also provide role-
4 independent information such as the IP addresses associated with the target system
5 and the OS version of the target system.

6 The procedures discussed below with respect to Figs. 6, 7 and 8 are
7 examples of procedures that may be performed by user interface application 110
8 (Fig. 1) using information contained in Runtime XML 108.

9 Fig. 6 is a flow diagram illustrating an embodiment of a procedure 600 for
10 masking role details from a user unless a user requests to see the role details.
11 Initially, procedure 600 identifies one or more roles associated with a target server
12 (block 602). For example, these roles may be identified in a Runtime XML file
13 received by the user interface application. The procedure then displays a list of the
14 identified roles to the user (block 604). This listing may include the name of the
15 target server (if any) and the roles that are associated with the target server. At this
16 point, the procedure masks (e.g., does not display) details regarding each of the
17 roles associated with the target server. The user may request details regarding one
18 or more of the listed roles (block 606). For example, the user may use a mouse or
19 other pointing device to select a particular role from the list of roles.

20 If the user requests details regarding a role, the procedure displays details
21 regarding the requested role (block 608). The retrieved details are displayed to the
22 user in the proximity of the associated role. Example details regarding a role
23 include a description of the functions performed by the role, dependencies on
24 other roles, services needed for the role to operate properly, the proper startup
25

1 mode for such services (automatic or manual), ports needed for the role to operate
2 properly, and the communication direction (inbound or outbound) for such ports.

3 In a particular embodiment, roles that are not relevant to the target server
4 are, by default, not presented to the user. However, those roles can be added by
5 the user, if desired. If a particular task depends on a role, that task is not displayed
6 to the user unless the corresponding role is selected. For example, if TaskA
7 depends on RoleA, the system will not display TaskA to the user unless RoleA is
8 selected. In various situations, a role may be dependent on another role and a
9 service may be dependent on another service. In any of these situations, if a user
10 selects or deselects a particular role or service, the dependent role or service is also
11 selected or deselected.

12 In an alternate embodiment, a procedure retrieves all details regarding
13 roles, tasks, services and ports for a particular server. However, only certain
14 details are shown to the user, unless the user requests to see additional details. The
15 particular details shown to the user are based on one or more roles selected by the
16 user.

17 Fig. 7 is a flow diagram illustrating an embodiment of a procedure 700 for
18 allowing a user to select services and ports associated with a particular role.
19 Initially, procedure 700 identifies a role associated with a target server (block
20 702). The procedure then identifies services associated with the role and displays
21 the identified services to the user (block 704). Since the procedure knows the role
22 associated with the target server, the list of services is limited to those that are
23 associated with the role, thereby focusing the list of services to those that are
24 relevant to the target server. Next, the selected services (i.e., services associated
25 with the role) are activated and the unselected services (i.e., services that are not

1 associated with the role) are deactivated (block 706). This “activation” and
2 “deactivation” of services will be identified in the Output XML file generated by
3 the user interface application, as discussed below. Further, the procedure
4 deactivates services not specified by the role (block 708).

5 Procedure 700 continues by identifying ports associated with the role (i.e.,
6 the role identified in block 702) and displaying the identified ports to the user
7 (block 710). Since the procedure knows the role associated with the target server,
8 the list of ports is limited to those that are associated with the role, thereby
9 focusing the list of ports to those that are relevant to the target server. In a
10 particular embodiment, block 710 is performed if the user has selected IPsec
11 filtering. The user is then asked to select one or more desired ports from among
12 the listed ports (block 712). This allows the user to make the final selection
13 regarding the ports that will be utilized on the target server. After the user has
14 selected one or more ports from the list of ports, the selected ports are activated
15 and the unselected ports are deactivated (block 714). Finally, an Output XML file
16 is created (block 716) that contains information used by the configuration engine
17 to configure the target server. Information contained in the Output XML file
18 includes data regarding activated and deactivated services and ports.

19 The end result of the various selections made in Fig. 7 is the definition of a
20 data packet filtering policy for the target server. By selecting available ports and
21 services, the user has effectively defined what types of data will be accepted by the
22 target server and what types of data will be rejected by the target server.

23 Additionally, by specifying exemptions to the packet filtering policy, the user can
24 effectively define what data sources will be permitted to communicate with the
25

1 target server and what data sources will not be permitted to communicate with the
2 target server.

3 Although the procedure shown in Fig. 7 specifically discusses services and
4 ports associated with a particular role, various other information can be made
5 available to the user for selection in configuring the target server. For example, a
6 user can filter data based on its source or destination address. Additionally, a user
7 can select one set of ports to allow on one interface and another set of ports to
8 allow on another interface. A user may also specify that traffic with certain
9 machines must be secured using encryption or integrity algorithms and may
10 specify that this traffic is exempt from previously established packet filtering
11 policies. Other information may be solicited from the user, such as
12 communication requirements for downlevel clients and audit objectives in order to
13 determine appropriate settings for security-relevant registry values as well as audit
14 policies.

15 Fig. 8 is a flow diagram illustrating an embodiment of a procedure 800 for
16 determining how to handle services and/or ports associated with a target server
17 that are not defined in a knowledge base. Initially, procedure 800 identifies all
18 services and ports associated with a target server (block 802). The procedure then
19 identifies specific services and/or ports associated with a target server that are not
20 defined in a knowledge base (block 804). For example, a particular service or port
21 on the target server does not have a corresponding entry in the knowledge base
22 defining the service or port. In this situation, neither the pre-processor nor the user
23 interface application has any knowledge of how the service or port operates or
24 when the service or port is needed.
25

1 The user is presented with a list of options for handling the specific services
2 and/or ports that are not defined in the knowledge base (block 806). The options
3 may include deactivating the services and/or ports or defining the operation of the
4 services and/or ports. The procedure receives the user's responses regarding
5 handling the specific services and/or ports and records the user's responses (block
6 808). These responses are used by the user interface application when generating
7 the Output XML file.

8 Certain details regarding the configuration of the target server may not be
9 contained in the knowledge base or known to the user configuring the target
10 server. For example, a user may not know the addresses of the target server's
11 domain controllers or DNS servers. However, these addresses can be determined
12 (e.g., from the active directory and the DNS servers) at the time the target server is
13 configured. Thus, the necessary information is obtained and set when the target
14 server is actually configured. Therefore, it is not necessary to have all details
15 regarding a target server until the target server is ready to be configured. This
16 allows the same abstract policy to be delivered and interpreted differently on
17 different machines.

18 In one embodiment, Internet Protocol Security (IPSec) is used by the
19 systems and methods discussed herein to assist in configuring the target server.
20 IPSec is a set of protocols that support secure exchange of data (e.g., data packets)
21 at the IP layer. Alternate embodiments may use other security protocols, such as
22 ICF (Internet Connection Firewall) or TCP/IP filters.

23 In a particular embodiment, certain ports and or services may not be
24 contained in the knowledge base. In this situation, the pre-processor (e.g., pre-
25 processor 102 in Fig. 1) is responsible for populating the <Ports> section of the

1 <Unknown> <Services> <Service> section with any ports that an unknown
2 service is listening on. For example:

```
3     <Unknown>
4     <Services>
5         <Service>
6             <Name> </Name>
7             ...
8             <Ports>
9                 <Port>
10                     <Name> </Name>
11                     <Type> </Type>
12                 </Port>
13             </Ports>
14         </Service>
15     </Services>
16 </Unknown>
```

13 This capability for the pre-processor uses a remotable interface (referred to as
14 “ActiveSocket”) to populate the <Ports> associated with an unknown service as
15 follows:

```
16
17     If ActiveSocket() call fails, then
18         Leave <Ports> section empty
19     Else {ActiveSocket was called and returned successfully}
20     If Service was unambiguously found to be listening on a port then:
21         If port number is already defined in the KB (under
22         \<Ports><Port><Number>)then
23             <Unknown>...<Ports><Port><Name> = existing
24         \<Ports><Port><Name>
25             Else {port was not found in the KB}
26                 If port < 1024, then
27                     //Update the Unknown Service Information
28                     <Unknown>...<Ports><Port><Name> = Port#
29                     service was listening on
30                     <Unknown>...<Ports><Port><Type> =
31                     Inbound
```

```

1      //Update the port definition section
2      \<Ports><Port><Name> = Port# service was
3      listening on
4      \<Ports><Port><Number> = Port# service was
5      listening on
6      \<Ports><Port><Protocols><Protocol><Name>
7      = TCP, UDP or Both
8      //Update the port localization section
9      Set ServiceName = the
10     \<ServiceLocalization><Service><DisplayName>
11     associated with <Unknown>...<Service><Name>
12     \<PortLocalization><Port><Name> = Port#
13     service was listening on.
14     \<PortLocalization><Port><Type> = Inbound
15     \<PortLocalization><Port><DisplayName> =
16     %ServiceName%
17     \<PortLocalization><Port><Name><Description>
18     n> = SCW determined that the %ServiceName%
19     service was listening on port
20     %\<Ports><Port><Number>%.
21     Else {port is >= 1024}
22     <Unknown>...<Ports><Port><Name> =
23     RANDOM
24     <Unknown>...<Ports><Port><Type> =
25     Inbound
26     Endif
27     Endif
28     Else {Service was not found to be listening exclusively on a port}
29     Leave <Ports> section empty
30     Endif
31 Endif

```

Thus, the pre-processor dynamically extends the knowledge base and provides this extra target server information to the User interface via the runtime XML. In a particular embodiment, if a service belongs to a process that is hosting other services, then the system does not determine who is listening to what and the system does not make any assumptions in this regard. The process is not concerned with loopback address 127.0.0.1, and treats all UDP ports as “listening”

1 (as long as they are not on the loopback address). TCP ports that are in the
2 listening state are treated as “listening”.

3 A sequence of “pages” or requests for information may be displayed on a
4 display device viewed by the user configuring one or more servers. This sequence
5 of pages request that the user identify functional and deployment requirements,
6 based on the knowledge base data and current disposition of the target server, in
7 order to configure the server. The information requested by the user (i.e., the
8 information contained on the pages) may vary depending on the user’s previous
9 selections and information contained in the knowledge base. Examples of the
10 various pages that may be used to collect information from the user are provided
11 below.

12
13 Select Server Roles – allows the user to select one or more server roles that
14 the target server will perform, such as DNS Server, File Server, etc.

15 Select Client Roles – allows the user to select one or more client roles, such
16 as DHCP Client or DNS Client.

17 Select Server Tasks – allows the user to select one or more tasks that are to
18 be performed by the server, such as collecting performance data, browsing the
19 network for computers and collecting help and support data.

20 Select Additional Services – allows the user to enable additional services,
21 such as 3rd Party Anti-virus or Backup services.

22 Handling Unspecified Services – allows the user to select, for example,
23 between 1) don’t change services that are not specified by this policy, and 2)
24 disable services that are not specified by this policy.

1 Configuring IP Security – allows the user to select whether they want to
2 activate the IP Security policy.

3 IP Security Usage – allows the user to select how they want to use IP
4 Security, such as securing network traffic with trusted computers or blocking
5 specific types of network traffic.

6 Packet Filtering Strategy – allows the user to select the basic packet
7 filtering strategy, such as filtering traffic over specific local interfaces or filtering
8 traffic from specific computers or subnets.

9 Open Ports – allows the user to select inbound or outbound ports to open in
10 order to allow network traffic, such as Port 20 (FTP-Data) and Port 80 (Web).
11 Since outbound requests normally require an inbound response, the system and
12 methods described herein automatically establish appropriate inbound block filters
13 to mitigate security holes created by allowing the inbound responses.

14 Allowing Traffic with Trusted Computers and Subnets – allows the user to
15 select trusted computers and subnets as exemptions to a previously established
16 packet filtering policy. Trusted computers can be explicitly identified (e.g. by
17 address) or implicitly identified (e.g. “Domain Controllers”).

18 Filter Traffic with Untrusted Computers and Subnets – allows the user to
19 filter traffic with specific remote computers or subnets.

20 Authentication Methods – allows the user to indicate how to establish trust
21 between computers, such as kerberos, certificates or a pre-shared key.

22 IP Security Summary – allows the user to review IP Security settings.

23 Configuring Registry Settings – allows the user to select whether they want
24 to activate a registry setting policy.
25

1 Require SMB Security Signatures – allows the user to indicate whether
2 Server Message Block (SMB) security signatures are enabled or required.

3 Require LDAP Signing – allows the user to determine whether LDAP
4 signing is required.

5 Outbound Authentication Methods – allows the user to determine the LAN
6 Manager authentication level used when making outbound connections.

7 Outbound Authentication using Domain Accounts – allows the user to
8 provide information about the domain controllers that contain the domain accounts
9 used to connect to other computers.

10 Outbound Authentication using Local Accounts – allows the user to provide
11 information about the target computer(s).

12 Inbound Authentication Methods – allows the user to provide information
13 about client computer(s) that connect to the target server or to the domain.

14 Registry Settings Summary – allows the user to review various registry
15 settings.

16
17 The above pages are provided for purposes of explanation. A particular
18 system may include all or a portion of the pages discussed above. Alternatively, a
19 particular system may include pages in addition to those discussed above. The
20 content of certain pages (e.g., the selections offered to the user) may vary based on
21 selections on other pages, information contained in one or more knowledge bases,
22 information regarding the target server(s), etc.

23 Figs. 9A and 9B illustrate an example of an Output XML file 900
24 associated with one or more servers. “Service Names” are the services that make
25 up a particular role. For example, a Web Server Role needs the HTTPfilter,

1 IISAdmin, and W3SVC services. In the example of Figs. 9A and 9B, only a few
2 Service Names are shown. A particular Output XML file may include any number
3 of Service Names.

4 Based on the roles that are selected, the corresponding Service Names are
5 enabled and the remaining services are disabled, thereby improving the security of
6 the system. Since each active service provides a potential area for security
7 failures, reducing the number of active services enhances security. "Extension
8 IDs" inform the configuration engine what extension understands the data
9 contained therein. For example, IPSec data can be passed to the IPSec extension
10 for processing. "Filters" define the interface on which the packet filtering process
11 will be implemented. "Port exemptions" and "Dynamic exemptions" identify the
12 traffic that is allowed through the associated filter.

13 The following is an example of a transformed version of the Output XML
14 file. This transformed version is generated by transformation module 118 and
15 applied to a target server.

```
16 ipsec
17 static
18 set store location=local
19
20 delete rule all "SCW Policy"
21
22 delete policy name="SCW Policy"
23
24 delete filteraction name="$SCW$_Block"
25 delete filteraction name="$SCW$_Permit"
delete filteraction name="$SCW$_Require-Hi"
delete filteraction name="$SCW$_Require-Lo"
delete filteraction name="$SCW$_Request-Hi"
delete filteraction name="$SCW$_Request-Lo"
26
delete filterlist name="$SCW$_DefaultDeny"
```

```

1 delete filterlist name="$SCW$_InboundPortExemptions"
2 delete filterlist name="$SCW$_OutboundPortExemptions"
3 delete filterlist name="$SCW$_InboundAttackVector"
4 delete filterlist name="$SCW$_MachineExemptions"
5 delete filterlist name="$SCW$_TrustedGroup1"
6 delete filterlist name="$SCW$_TrustedGroup2"
7 delete filterlist name="$SCW$_TrustedGroup3"
8 delete filterlist name="$SCW$_TrustedGroup4"
9 delete filterlist name="$SCW$_DynamicExemptions"
10 delete filterlist name="$SCW$_ProtocolExemptions"
11
12 add policy name="SCW Policy" description="Security Configuration
13 Wizard Policy" activatedefaultrule=no assign=no
14
15 add filteraction name="$SCW$_Block" description="Discard"
16 action=block
17 add filteraction name="$SCW$_Permit" description="Pass without
18 modification or security" action=permit
19 add filteraction name="$SCW$_Require-Hi" description="Negotiate, No
20 Fallback, No Inpass" inpass=no soft=No action=negotiate qmsec="AH[SHA1]+
21 ESP[None,SHA1] AH[MD5]+ESP[None,SHA1] AH[SHA1]+ESP[None,MD5]
22 AH[MD5]+ESP[None,MD5] ESP[None,SHA1] ESP[None,MD5] AH[SHA1]
23 AH[MD5] "
24 add filteraction name="$SCW$_Require-Lo" description="Negotiate, No
25 Fallback, Inpass" inpass=yes soft=No action=negotiate qmsec="AH[SHA1]+
26 ESP[None,SHA1] AH[MD5]+ESP[None,SHA1] AH[SHA1]+ESP[None,MD5]
27 AH[MD5]+ESP[None,MD5] ESP[None,SHA1] ESP[None,MD5] AH[SHA1]
28 AH[MD5] "
29 add filteraction name="$SCW$_Request-Hi" description="Negotiate,
30 Fallback, No Inpass" inpass=no soft=Yes action=negotiate qmsec="AH[SHA1]+
31 ESP[None,SHA1] AH[MD5]+ESP[None,SHA1] AH[SHA1]+ESP[None,MD5]
32 AH[MD5]+ESP[None,MD5] ESP[None,SHA1] ESP[None,MD5] AH[SHA1]
33 AH[MD5] "
34 add filteraction name="$SCW$_Request-Lo" description="Negotiate,
35 Fallback, Inpass" inpass=yes soft=Yes action=negotiate qmsec="AH[SHA1]+
36 ESP[None,SHA1] AH[MD5]+ESP[None,SHA1] AH[SHA1]+ESP[None,MD5]
37 AH[MD5]+ESP[None,MD5] ESP[None,SHA1] ESP[None,MD5] AH[SHA1]
38 AH[MD5] "
39
40 add filterlist name="$SCW$_DefaultDeny" description="Traffic to be
41 blocked"

```

1 add filterlist name="\$SCW\$_InboundPortExemptions" description="Ports
to permit (Inbound)"

2 add filterlist name="\$SCW\$_OutboundPortExemptions"
3 description="Ports to permit (Outbound)"

4 add filterlist name="\$SCW\$_InboundAttackVector" description="Mitigate
5 holes opened by outbound communication requirement"

6 add filterlist name="\$SCW\$_MachineExemptions" description="Machines
or Subnets to Permit"

7 add filterlist name="\$SCW\$_TrustedGroup1" description="Machines or
8 Subnets to Require Security with Hi"

9 add filterlist name="\$SCW\$_TrustedGroup2" description="Machines or
10 Subnets to Require Security with Lo"

11 add filterlist name="\$SCW\$_TrustedGroup3" description="Machines or
12 Subnets to Request Security with Hi"

13 add filterlist name="\$SCW\$_TrustedGroup4" description="Machines or
Subnets to Request Security with Lo"

14 add filterlist name="\$SCW\$_DynamicExemptions"
15 description="Infrastructure Traffic"

16 add filterlist name="\$SCW\$_ProtocolExemptions" description="IP
17 Protocols to Permit"

18 add filter filterlist="\$SCW\$_DefaultDeny" dstaddr=157.59.132.0
srcaddr=any description="any to me/IP" protocol=ANY mirrored=yes
19 srcmask=255.255.255.255 dstmask=255.255.252.0

20 add filter filterlist="\$SCW\$_InboundPortExemptions"
dstaddr=157.59.132.0 srcaddr=any description="any to me/IP" protocol=TCP
21 mirrored=yes srcmask=255.255.255.255 dstmask=255.255.252.0 dstport=80
22 srcport=0

23 add filter filterlist="\$SCW\$_InboundPortExemptions"
dstaddr=157.59.132.0 srcaddr=any description="any to me/IP" protocol=TCP
24 mirrored=yes srcmask=255.255.255.255 dstmask=255.255.252.0 dstport=443
25 srcport=0

1 add filter filterlist="\$SCW\$_ProtocolExemptions" dstaddr=me srcaddr=any
2 description="Protocol filter" protocol=ICMP mirrored=yes
3 srcmask=255.255.255.255 dstmask=255.255.255.255

4 add rule name="\$SCW\$_Block_DefaultDeny" policy="SCW Policy"
5 filterlist="\$SCW\$_DefaultDeny" filteraction="\$SCW\$_Block" activate=yes

6 add rule name="\$SCW\$_Permit_InboundPortExemptions" policy="SCW
7 Policy" filterlist="\$SCW\$_InboundPortExemptions" filteraction=
8 "\$SCW\$_Permit" activate=yes

9 add rule name="\$SCW\$_Permit_OutboundPortExemptions" policy="SCW
10 Policy" filterlist="\$SCW\$_OutboundPortExemptions"
11 filteraction="\$SCW\$_Permit" activate=yes

12 add rule name="\$SCW\$_MitigateInboundAttackVector" policy="SCW
13 Policy" filterlist="\$SCW\$_InboundAttackVector" filteraction="\$SCW\$_Block"
14 activate=yes

15 add rule name="\$SCW\$_Permit_MachineExemptions" policy="SCW
16 Policy" filterlist="\$SCW\$_MachineExemptions" filteraction="\$SCW\$_Permit"
17 activate=yes

18 add rule name="\$SCW\$_Require-Hi_TrustedGroup1" policy="SCW
19 Policy" filterlist="\$SCW\$_TrustedGroup1" filteraction="\$SCW\$_Require-Hi"
20 activate=yes kerberos=Yes

21 add rule name="\$SCW\$_Require-Lo_TrustedGroup2" policy="SCW
22 Policy" filterlist="\$SCW\$_TrustedGroup2" filteraction="\$SCW\$_Require-Lo"
23 activate=yes kerberos=Yes

24 add rule name="\$SCW\$_Request-Hi_TrustedGroup3" policy="SCW
25 Policy" filterlist="\$SCW\$_TrustedGroup3" filteraction="\$SCW\$_Request-Hi"
26 activate=yes kerberos=Yes

27 add rule name="\$SCW\$_Request-Lo_TrustedGroup4" policy="SCW
28 Policy" filterlist="\$SCW\$_TrustedGroup4" filteraction="\$SCW\$_Request-Lo"
29 activate=yes kerberos=Yes

30 add rule name="\$SCW\$_Permit_DynamicExemptions" policy="SCW
31 Policy" filterlist="\$SCW\$_DynamicExemptions" filteraction="\$SCW\$_Permit"
32 activate=yes

1 add rule name="\$SCW\$_Permit_ProtocolExemptions" policy="SCW
2 Policy" filterlist="\$SCW\$_ProtocolExemptions" filteraction="\$SCW\$_Permit"
activate=yes

3 set policy name="SCW Policy" assign=y
4
5

6 The above transformed version of the Output XML file contains various settings
7 and other information used to configure the target server. An IPsec filter describes
8 a subset of network traffic in terms of five parameters: SA (Source Address), DA
9 (Destination Address), SP (Source Port), DP (Destination Port), and Protocol. A
10 Filter List includes one or more such filters that may describe a larger subset of
11 traffic that can trigger a specific filter action, such as Block or Permit. Rules
12 associate filter lists with filter actions.

13 Fig. 10 illustrates an example of a computing environment 1000 within
14 which the server configuration systems and methods, can be either fully or
15 partially implemented. Computing environment 1000 is only one example of a
16 computing system and is not intended to suggest any limitation as to the scope of
17 use or functionality of the network architectures. Neither should the computing
18 environment 1000 be interpreted as having any dependency or requirement
19 relating to any one or combination of components illustrated in the example
20 computing environment 1000.

21 The computer and network architectures can be implemented with
22 numerous other general purpose or special purpose computing system
23 environments or configurations. Examples of well known computing systems,
24 environments, and/or configurations that may be suitable for use include, but are
25 not limited to, personal computers, server computers, thin clients, thick clients,

1 hand-held or laptop devices, multiprocessor systems, microprocessor-based
2 systems, set top boxes, programmable consumer electronics, network PCs,
3 minicomputers, mainframe computers, gaming consoles, distributed computing
4 environments that include any of the above systems or devices, and the like.

5 The computing environment 1000 includes a general-purpose computing
6 system in the form of a computing device 1002. The components of computing
7 device 1002 can include, but are not limited to, one or more processors 1004 (e.g.,
8 any of microprocessors, controllers, and the like), a system memory 1006, and a
9 system bus 1008 that couples various system components including the processor
10 1004 to the system memory 1006. The one or more processors 1004 process
11 various computer-executable instructions to control the operation of computing
12 device 1002 and to communicate with other electronic and computing devices.

13 The system bus 1008 represents any number of several types of bus or
14 switching structures, including a memory bus or memory controller, point-to-point
15 connections, a switching fabric, a peripheral bus, an accelerated graphics port, and
16 a processor or local bus using any of a variety of bus architectures. By way of
17 example, such architectures can include an Industry Standard Architecture (ISA)
18 bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a
19 Video Electronics Standards Association (VESA) local bus, and a Peripheral
20 Component Interconnects (PCI) bus also known as a Mezzanine bus.

21 Computing environment 1000 typically includes a variety of computer-
22 readable media. Such media can be any available media that is accessible by
23 computing device 1002 and includes both volatile and non-volatile media,
24 removable and non-removable media. The system memory 1006 includes
25 computer-readable media in the form of volatile memory, such as random access

1 memory (RAM) 1010, and/or non-volatile memory, such as read only memory
2 (ROM) 1012. A basic input/output system (BIOS) 1014, containing the basic
3 routines that help to transfer information between elements within computing
4 device 1002, such as during start-up, is stored in ROM 1012. RAM 1010 typically
5 contains data and/or program modules that are immediately accessible to and/or
6 presently operated on by the processing unit 1004.

7 Computing device 1002 can also include other removable/non-removable,
8 volatile/non-volatile computer storage media. By way of example, a hard disk
9 drive 1016 is included for reading from and writing to a non-removable, non-
10 volatile magnetic media (not shown), a magnetic disk drive 1018 for reading from
11 and writing to a removable, non-volatile magnetic disk 1020 (e.g., a “floppy
12 disk”), and an optical disk drive 1022 for reading from and/or writing to a
13 removable, non-volatile optical disk 1024 such as a CD-ROM, DVD, or any other
14 type of optical media. The hard disk drive 1016, magnetic disk drive 1018, and
15 optical disk drive 1022 are each connected to the system bus 1008 by one or more
16 data media interfaces 1026. Alternatively, the hard disk drive 1016, magnetic disk
17 drive 1018, and optical disk drive 1022 can be connected to the system bus 1008
18 by a SCSI interface (not shown).

19 The disk drives and their associated computer-readable media provide
20 non-volatile storage of computer-readable instructions, data structures, program
21 modules, and other data for computing device 1002. Although the example
22 illustrates a hard disk 1016, a removable magnetic disk 1020, and a removable
23 optical disk 1024, it is to be appreciated that other types of computer-readable
24 media which can store data that is accessible by a computer, such as magnetic
25 cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital

1 versatile disks (DVD) or other optical storage, random access memories (RAM),
2 read only memories (ROM), electrically erasable programmable read-only
3 memory (EEPROM), and the like, can also be utilized to implement the example
4 computing system and environment.

5 Any number of program modules can be stored on the hard disk 1016,
6 magnetic disk 1020, optical disk 1024, ROM 1012, and/or RAM 1010, including
7 by way of example, an operating system 1026, one or more application programs
8 1028, other program modules 1030, and program data 1032. Each of such
9 operating system 1026, one or more application programs 1028, other program
10 modules 1030, and program data 1032 (or some combination thereof) may include
11 an embodiment of the systems and methods for a test instantiation system.

12 Computing device 1002 can include a variety of computer-readable media
13 identified as communication media. Communication media typically embodies
14 computer-readable instructions, data structures, program modules, or other data in
15 a modulated data signal such as a carrier wave or other transport mechanism and
16 includes any information delivery media. The term “modulated data signal” refers
17 to a signal that has one or more of its characteristics set or changed in such a
18 manner as to encode information in the signal. By way of example, and not
19 limitation, communication media includes wired media such as a wired network or
20 direct-wired connection, and wireless media such as acoustic, RF, infrared, and
21 other wireless media. Combinations of any of the above are also included within
22 the scope of computer-readable media.

23 A user can enter commands and information into computing device 1002
24 via input devices such as a keyboard 1034 and a pointing device 1036 (e.g., a
25 “mouse”). Other input devices 1038 (not shown specifically) may include a

1 microphone, joystick, game pad, controller, satellite dish, serial port, scanner,
2 and/or the like. These and other input devices are connected to the processing unit
3 1004 via input/output interfaces 1040 that are coupled to the system bus 1008, but
4 may be connected by other interface and bus structures, such as a parallel port,
5 game port, and/or a universal serial bus (USB).

6 A monitor 1042 or other type of display device can also be connected to the
7 system bus 1008 via an interface, such as a video adapter 1044. In addition to the
8 monitor 1042, other output peripheral devices can include components such as
9 speakers (not shown) and a printer 1046 which can be connected to computing
10 device 1002 via the input/output interfaces 1040.

11 Computing device 1002 can operate in a networked environment using
12 logical connections to one or more remote computers, such as a remote computing
13 device 1048. By way of example, the remote computing device 1048 can be a
14 personal computer, portable computer, a server, a router, a network computer, a
15 peer device or other common network node, and the like. The remote computing
16 device 1048 is illustrated as a portable computer that can include many or all of
17 the elements and features described herein relative to computing device 1002.

18 Logical connections between computing device 1002 and the remote
19 computer 1048 are depicted as a local area network (LAN) 1050 and a general
20 wide area network (WAN) 1052. Such networking environments are
21 commonplace in offices, enterprise-wide computer networks, intranets, and the
22 Internet. When implemented in a LAN networking environment, the computing
23 device 1002 is connected to a local network 1050 via a network interface or
24 adapter 1054. When implemented in a WAN networking environment, the
25 computing device 1002 typically includes a modem 1056 or other means for

1 establishing communications over the wide network 1052. The modem 1056,
2 which can be internal or external to computing device 1002, can be connected to
3 the system bus 1008 via the input/output interfaces 1040 or other appropriate
4 mechanisms. It is to be appreciated that the illustrated network connections are
5 exemplary and that other means of establishing communication link(s) between
6 the computing devices 1002 and 1048 can be employed.

7 In a networked environment, such as that illustrated with computing
8 environment 1000, program modules depicted relative to the computing device
9 1002, or portions thereof, may be stored in a remote memory storage device. By
10 way of example, remote application programs 1058 reside on a memory device of
11 remote computing device 1048. For purposes of illustration, application programs
12 and other executable program components, such as the operating system, are
13 illustrated herein as discrete blocks, although it is recognized that such programs
14 and components reside at various times in different storage components of the
15 computer system 1002, and are executed by the data processor(s) of the computer.

16 Particular examples discussed herein relate to creating a security policy and
17 configuring a server with that security policy. Similar procedures can be used to
18 modify existing security policies, apply an existing security policy to another
19 server, or analyze a server for compliance with a particular security policy.

20 Although the description above uses language that is specific to structural
21 features and/or methodological acts, it is to be understood that the invention
22 defined in the appended claims is not limited to the specific features or acts
23 described. Rather, the specific features and acts are disclosed as exemplary forms
24 of implementing the invention.
25